UNIVERSIDAD AUTÓNOMA DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

Doble Grado en Ingeniería Informática y Matemáticas

TRABAJO FIN DE GRADO

ALGORITMOS CUÁNTICOS PARA LA RESOLUCIÓN DEL PROBLEMA DE SATISFACIBILIDAD BOOLEANA

María Prieto Gil
Tutor: Estrella Pulido Cañabate

Junio 2019

ALGORITMOS CUÁNTICOS PARA LA RESOLUCIÓN DEL PROBLEMA DE SATISFACIBILIDAD BOOLEANA

AUTOR: María Prieto Gil
TUTOR: Estrella Pulido Cañabate

Escuela Politécnica Superior
Universidad Autónoma de Madrid
Junio de 2019

Resumen

Este Trabajo Fin de Grado tiene como objetivo el estudio de la computación cuántica y el desarrollo de un algoritmo cuántico para la resolución de una variante del problema de satisfacibilidad booleana.

La computación cuántica es una rama de investigación que está actualmente en sus inicios. Basa su funcionamiento en fenómenos de la física cuántica, que permite realizar cálculos de forma paralelizable. El interés por este campo surge por la capacidad que podrían alcanzar los ordenadores cuánticos de resolver problemas computacionalmente más rápido que los ordenadores clásicos.

Tras la comprensión de los principios básicos de la computación cuántica se han estudiado en detalle algunos de los algoritmos cuánticos más conocidos, centrándose en el algoritmo de Grover.

Finalmente se ha podido llevar a cabo el desarrollo de un algoritmo para resolver una variante del problema de satisfacibilidad booleana, el problema 'Exactly-1 3-SAT', que basa su funcionamiento en el algoritmo de Grover. Se han desarrollado tres posibles implementaciones de este algoritmo y se han realizado pruebas tanto en un simulador, como en un ordenador cuántico real.

El problema 'Exactly-1 3-SAT' es un problema perteneciente a la clase de complejidad NP-completo, cuya resolución de forma cuántica reduce el tiempo de computación notablemente con respecto a la forma clásica.

Palabras clave

Ordenador cuántico, qubit, algoritmo de Grover, problema de satisfacibilidad booleana.

Abstract

This Final Degree Project aims at the study of quantum computing and the development of a quantum algorithm for the resolution of the boolean satisfaction problem variant.

Quantum computing is a branch of research that is currently in its early stages of development. It bases its operation on quantum physics phenomena, which allows calculations to be made in a parallel way. Interest in this field arises because of the ability of quantum computers to solve problems computationally faster than classical computers.

After understanding the basic principles of quantum computing, some of the best-known quantum algorithms have been studied in detail, focusing on the Grover algorithm.

Finally, it has been possible to develop an algorithm to solve a variant of the Boolean satisfaction problem, the problem 'Exactly-1 3-SAT', which bases its operation on the Grover algorithm. Three possible implementations of this algorithm have been developed and tests have been carried out both in a simulator and in a real quantum computer.

The problem 'Exactly-1 3-SAT' is a problem belonging to the class of complexity NP-complete, whose resolution of quantum form reduces the computing time notably with respect to the classical form.

Keywords

Quantum computer, qubit, Grover algorithm, Boolean satisfiability problem.

Agradecimientos

Son muchas las personas me han ayudado a poder completar con éxito este Trabajo de Fin de Grado. En primer lugar, muchas gracias a Estrella por ofrecerme la oportunidad de trabajar en este proyecto y ofrecerme todas las facilidades para llevarlo a cabo. Trabajar con ella ha sido todo un placer.

También gracias a Gonzalo, que aun siendo un tema tan nuevo para todos se ha esforzado en ayudarme en todo lo que podía, aportando ideas cuando ya no sabía qué hacer.

Gracias al equipo de desarrollo de Qiskit, en especial a Juan Gómez, por ofrecerme su ayuda e informarme de cualquier novedad en sus dispositivos cuánticos.

Y, por último, gracias a mis padres por su paciencia, cariño y apoyo incondicional en todo lo que hago.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1 Introducción

1.1 Motivación

Este Trabajo de Fin de Grado está motivado por el auge de la computación cuántica en los últimos años. Esta nueva forma de computación prácticamente acaba de nacer y cualquier contribución a su desarrollo es importante. Actualmente son varias las empresas que compiten por conseguir un ordenador cuántico que consiga realizar cualquier tarea computacionalmente más rápido que un ordenador clásico.

El interés por este campo surge por la capacidad que podrían alcanzar los ordenadores cuánticos de resolver problemas en mucho menos tiempo que los ordenadores clásicos. Un ejemplo de esto es el problema de factorizar números enteros. Peter Shor propuso un algoritmo cuántico que factorizaba números enteros en tiempo polinómico. Este descubrimiento disparó el interés en la computación cuántica y se espera que pronto estos sistemas sean capaces de descomponer números suficientemente grandes para quebrantar los múltiples sistemas criptográficos que se basan en esta dificultad de descomposición.

La intención de este trabajo es la comprensión de las ideas principales de la computación cuántica y los algoritmos cuánticos, en especial el algoritmo de Grover. El objetivo es aplicar estos conocimientos para el desarrollo de un algoritmo que resuelva un problema perteneciente a la clase de complejidad NP-completo, en concreto, el *problema de satisfacibilidad booleana*.

1.2 Objetivos

El objetivo de este Trabajo de Fin de Grado es el diseño de un algoritmo cuántico para la resolución del **problema de satisfacibilidad booleana**. Para llevar acabo este trabajo se ha partido de un estudio del algoritmo de Grover, que es una de las partes principales del algoritmo desarrollado.

La idea inicial era realizar un algoritmo general, es decir, que fuese válido para cualquier entrada. Tras un profundo estudio se llegó a la conclusión de que esto no era posible, ya que depende totalmente de las características del ordenador cuántico en el que se ejecute. Por tanto, se ha desarrollado un algoritmo que resuelve el problema anteriormente mencionado con unas características de entrada específicas que se detallarán en el capítulo 4.

Con este trabajo se pretende contribuir al desarrollo de algoritmos cuánticos para conocer más acerca de su funcionamiento. Cuando la topología y estabilidad de estos ordenadores mejore, se podrá aplicar el problema a una entrada más general obteniéndose resultados correctos. La mayor parte de las pruebas realizadas para comprobar el funcionamiento se

han realizado usando un simulador, pero se espera que, en el futuro, cuando los ordenadores mejoren, también se obtengan resultados correctos en un ordenador cuántico real.

1.3 Organización de la memoria

La memoria está organizada en los siguientes capítulos:

- **Estado del arte:** en este capítulo se hace un repaso al estado en el que se encuentra en la actualidad la computación cuántica. También se explican las herramientas utilizadas para el desarrollo de este Trabajo de Fin de Grado

- **Computación cuántica:** en este capítulo se hace una introducción a los conceptos básicos de la computación cuántica necesarios para la comprensión del algoritmo desarrollado durante este trabajo.

- **Diseño:** se trata del capítulo principal de esta memoria. En él se explican los detalles de la implementación del algoritmo desarrollado.

- **Pruebas y resultados:** en este capítulo se describen las distintas pruebas realizadas para comprobar el correcto funcionamiento del algoritmo. También se incluyen gráficas y diagramas con los resultados obtenidos.

- **Conclusiones y trabajo futuro:** en este último capítulo se incluyen las distintas conclusiones extraídas, además se proponen posibles mejoras y trabajo futuro.

- **Anexos:** estos apartados contienen información extra que no es estrictamente necesaria para la comprensión del algoritmo desarrollado a lo largo de este Trabajo de Fin de Grado, pero se recomienda su lectura.

2 Estado del arte

2.1 Computación cuántica

La idea del ordenador cuántico se remonta a las décadas de los 70 y 80, cuando Richard Feynman, Paul Benioff y David Deutsche proponen el concepto de computación cuántica. Richard Feyman muestra cómo un sistema cuántico puede actuar como simulador para procesos cuánticos y además ser utilizado para mejorar el rendimiento computacional clásico [1].

En 1994, Peter Shor propuso el primer algoritmo cuántico, el **algoritmo de Shor**, cuyo éxito impulsó la búsqueda de un computador cuántico. En 1998, Chuang implementó el primer computador cuántico de un qubit y en 1999 Grover desarrolló un algoritmo para realizar búsquedas en listas utilizando 3 qubits.

Los ordenadores cuánticos codifican la información como una serie de estados mecánicos cuánticos y aprovechan la superposición cuántica para realizar diversos cálculos simultáneamente. Este es uno de los motivos por los que la computación cuántica obtiene resultados computacionalmente más rápidos que la computación clásica.

Actualmente, en el mundo de la computación cuántica, se vive en una analogía con los años 50, la época en la que se iniciaron los ordenadores que operaban con transistores y ocupaban grandes salas. No paran de surgir nuevas hipótesis y algoritmos para mejorar su funcionamiento, pero aún se presentan grandes retos como la *decoherencia*, el aumento del número de qubits, la lectura de información… [2]

2.2 Algoritmo desarrollado

Este Trabajo de Fin de Grado se basa en el desarrollo de un algoritmo cuántico para resolver el problema de satisfacibilidad booleana. Se trata de un problema NP-completo [5] cuya resolución de forma cuántica reduce el tiempo de computación notablemente, con respecto a la forma clásica.

Para la implementación de este algoritmo se ha realizado un profundo trabajo de investigación. Durante este proceso se encontró un artículo [3] que resolvía este problema, pero de forma errónea. Por este motivo, el trabajo se centró en resolver el error que se cometía al realizar el algoritmo con distintas alternativas, además de su ampliación y pruebas en los ordenadores cuánticos y simuladores que ofrece IBM.

2.3 Herramientas

Para la implementación del algoritmo diseñado en este Trabajo de Fin de Grado se ha utilizado el lenguaje de programación *Python*, específicamente el framework de código abierto ***Qiskit*** [4].

Qiskit es una herramienta desarrollada por el equipo de *IBM Research* que permite a los usuarios usar simuladores cuánticos y lanzar programas a ordenadores cuánticos reales. Actualmente se pueden utilizar cualquiera de los tres ordenadores disponibles, dos de ellos tienen 5 qubits y otro 14. La topología y detalles de estos ordenadores se puede consultar en el apartado 5.3.1. IBM prevé tener accesible para su uso en poco tiempo el ordenador de 20 qubits y están trabajando en uno de 50, posiblemente disponible a final de este año.

Qiskit está formado por cuatro componentes: *Terra, Aqua, Aer* e *Ignis. Terra* se utiliza para crear circuitos a bajo nivel, trabajando directamente con puertas cuánticas. *Aqua* proporciona herramientas que pueden usarse sin que el usuario tenga conocimientos de programación cuántica. Actualmente soporta aplicaciones en química, inteligencia artificial, optimización y finanzas. *Aer* proporciona simuladores de ordenadores cuánticos, permitiendo también la simulación del ruido. *Ignis* es un componente que contendrá herramientas para caracterizar el ruido en dispositivos a corto plazo.

Otra herramienta utilizada fue *Slack-Qiskit* [5], un foro online en el que se crea una comunidad que comparte información y dudas acerca de computación cuántica utilizando *Qiskit.*

También fueron útiles los conocimientos adquiridos y compartidos en la *Qiskit Hackathon Madrid*, organizada por *IBM Research* los días 25 y 26 de mayo. Durante estas jornadas, desarrolladores de Qiskit mostraron algunas herramientas que aún estaban integradas en la librería y fueron muy útiles para el desarrollo del algoritmo explicado en el capítulo 4.

3 Computación cuántica

En este apartado se hace una introducción a los conceptos básicos de la computación cuántica para facilitar la comprensión del algoritmo diseñado e implementado como parte de este trabajo.

3.1 Conceptos básicos

La computación cuántica utiliza las herramientas de la mecánica cuántica para su desarrollo. Por este motivo es necesario comprender sus principios más básicos. A continuación, se van a describir muy brevemente algunos de estos principios para que el lector obtenga una idea general de los mismos [6].

- **Superposición:** según este fenómeno, un elemento puede poseer dos o más valores simultáneamente.

- **Entrelazamiento:** fenómeno físico que se produce cuando un grupo de partículas interactúan de manera que el estado cuántico de cada partícula no se puede describir independientemente del estado de las otras, incluso cuando las partículas están separadas por una gran distancia.

- **Colapso de la función de onda:** ocurre cuando una función de onda, inicialmente en una superposición de varios estados, se reduce a un único estado debido a la interacción con el mundo externo. Esta es la esencia de la medición en mecánica cuántica.

Además de estos tres conceptos, también es importante conocer la unidad básica de la computación cuántica: el **qubit**. Un qubit en computación cuántica es el equivalente a un bit en computación clásica. La diferencia es que un qubit puede estar en el estado 0, 1 o una combinación lineal de estos dos estados, que es lo que llamamos superposición:

$$|\psi\rangle = \alpha|0\rangle + \beta\,|1\rangle, \quad \alpha, \beta \in \mathbb{C}$$

Esta notación es la estándar para definir estados cuánticos en mecánica cuántica. Se la conoce como *bra-ket* y es la que se utilizará a lo largo de este Trabajo de Fin de Grado.

Es importante comprender que en el momento en el que se mide el estado de un qubit se produce el fenómeno anteriormente mencionado de colapso de la función de onda por lo que este pierde la propiedad de superposición y solo podrá tomar los valores 0 o 1. En la fórmula anterior $|\alpha|^2$ es la probabilidad de que al medir el qubit tome el valor 0 y $|\beta|^2$ que tome el valor 1. Como las probabilidades deben sumar 1, se tiene que cumplir que $|\alpha|^2 + |\beta|^2 = 1$.

Tras esta breve introducción se puede definir el concepto de **ordenador cuántico**. Se trata de un ordenador que usa ciertos fenómenos de la mecánica cuántica (superposición y entrelazamiento) para realizar operaciones sobre los datos. Esta máquina está compuesta por qubits y su funcionamiento se basa en aplicarles ciertas *puertas cuánticas* y medir los resultados. A este conjunto de operaciones se le conoce como *algoritmo cuántico*.

3.2 Puertas cuánticas

En este apartado se describirán las puertas cuánticas utilizadas en el algoritmo desarrollado en este trabajo. Además, se describirán otras que no se han utilizado pero son importantes para los circuitos cuánticos.

Una **puerta cuántica** es, en computación cuántica, lo equivalente a una puerta lógica en computación clásica. Se puede diferenciar entre las *puertas unitarias*, que son aquellas que afectan a un solo qubit; y las *multiqubit*, que son las que afectan a varios qubits.

3.2.1 Puertas unitarias

Se pueden tratar las puertas como matrices unitarias (U tal que $U^*U = UU^* = I$, donde U^* es la matriz conjugada transpuesta). Por este motivo será útil escribir el qubit en forma vectorial:

$$|\psi\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$$

De esta forma, aplicar una puerta X a un qubit ψ, es equivalente a multiplicar X por ψ. Esto se explicará con más profundidad cuando se describan las puertas individualmente.

Para comprender mejor el funcionamiento de estas puertas, es útil introducir el concepto de **esfera de Bloch**. En mecánica cuántica es una representación geométrica de un estado. Como aparece en la figura 3-1, se puede utilizar para representar un qubit como un punto en esta esfera y observar cómo varía tras aplicarle alguna puerta unitaria.
Como se comentó en el apartado anterior, los estados básicos de un qubit son 0 y 1, representados como $|0\rangle$ y $|1\rangle$ en notación bra-ket. Cuando el punto se encuentra en el Norte de la esfera, el qubit tendrá valor $|0\rangle$; y cuando se encuentra en el Sur tendrá valor $|1\rangle$.

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

Se puede representar cualquier estado (punto en la esfera de Bloch) como una combinación lineal de $|0\rangle$ y $|1\rangle$:

$$|\psi\rangle = cos\left(\frac{\theta}{2}\right)|0\rangle + sen\left(\frac{\theta}{2}\right)e^{i\phi}|1\rangle$$

También es importante introducir otros dos estados: $|+\rangle$ y $|-\rangle$. Un qubit se encuentra en alguno de estos estados cuando está en superposición y la probabilidad de que al medirlo valga 1 es del 50% y de que valga 0 es también del 50%.

$$|+\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}} \quad |-\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}}$$

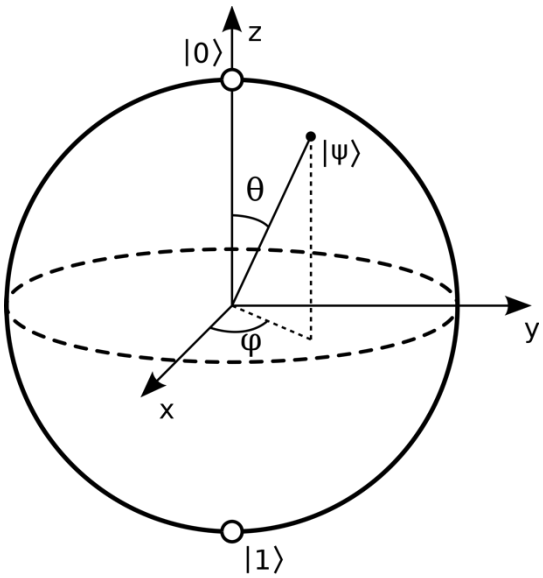


Figura 3-1: Representación de los estados básicos en la esfera de Bloch.

Tras esta introducción se pueden ver las puertas unitarias como transformaciones en la esfera de Bloch. Se pueden dividir en tres grupos: puertas de Pauli, puerta Hadamard y puertas de fase [7].

- **Puertas de Pauli**: Son aquellas que aplican al qubit un giro de ángulo π con respecto a los ejes X, Y, Z. Dependiendo del eje reciben el nombre de puertas **X**, **Y**, o **Z**. El efecto que causan sobre los estados básicos del qubit descritos anteriormente se pueden observar en la tabla 3.X.

	X	**Y**	**Z**
$\|0\rangle$	$\|1\rangle$	$\|1\rangle$	$\|0\rangle$
$\|1\rangle$	$\|0\rangle$	$\|0\rangle$	$\|1\rangle$
$\|+\rangle$	$\|+\rangle$	$\|-\rangle$	$\|-\rangle$
$\|-\rangle$	$\|-\rangle$	$\|+\rangle$	$\|+\rangle$
MATRIZ	$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$

Tabla 3-1: Resumen de las puertas de Pauli

- **Puerta Hadamard**: También conocida como puerta **H**. Se trata de una concatenación de un giro de ángulo $\frac{\pi}{2}$ con respecto al eje Y y otro de ángulo π con respecto al eje Z.

	H
$\|0\rangle$	$\|+\rangle$
$\|1\rangle$	$\|-\rangle$
$\|+\rangle$	$\|0\rangle$
$\|-\rangle$	$\|1\rangle$
MATRIZ	$\frac{1}{\sqrt{(2)}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}$

Tabla 3-2: Resumen de la puerta de Hadamard.

- **Puertas de fase:** Son giros sobre el eje Z. Estas puertas no modifican el valor final del qubit, pero sí afectan a futuras operaciones sobre el mismo. Dentro de este grupo se encuentran las puertas S y S^t, que corresponden a un giro de ángulo $\frac{\pi}{2}$ y de - $\frac{\pi}{2}$ respectivamente. También se encuentran las puertas T y T^t, que realizan un giro de ángulo de $\frac{\pi}{4}$ y - $\frac{\pi}{4}$ respectivamente.

	S	**S^t**	**T**	**T^t**
MATRIZ	$\begin{pmatrix}1 & 0\\ 0 & i\end{pmatrix}$	$\begin{pmatrix}1 & 0\\ 0 & -i\end{pmatrix}$	$\begin{pmatrix}1 & 0\\ 0 & e^{\frac{i\pi}{4}}\end{pmatrix}$	$\begin{pmatrix}1 & 0\\ 0 & e^{-\frac{i\pi}{4}}\end{pmatrix}$

Tabla 3-3: Resumen de las puertas de fase.

Todas las puertas unitarias que se acaban de detallar se convierten en las puertas U_1, U_2 y U_3 antes de ejecutarse en un ordenador cuántico real. Por este motivo se las conoce como *puertas físicas*. La más general es:

$$U_3(\theta,\phi,\lambda)=\begin{pmatrix}\cos(\theta/2) & -e^{i\lambda}\sin(\theta/2)\\ e^{i\phi}\sin(\theta/2) & e^{i\lambda+i\phi}\cos(\theta/2)\end{pmatrix}.$$

Esta puerta tiene el efecto de rotar un qubit que se encuentra en el estado inicial $|0\rangle$ a un estado arbitrario de superposición con la siguiente fase:

$$U_3|0\rangle=\cos(\theta/2)|0\rangle+\sin(\theta/2)e^{i\phi}|1\rangle$$

La puerta U_1 realiza un giro con respecto al eje Z. Es como las puertas de fase vistas anteriormente, pero con un ángulo general. La relación entre esta puerta y U_3 es la siguiente:

$$U_1(\lambda) = U_3(0,0,\lambda) = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\lambda} \end{pmatrix}$$

La puerta U_2 se utiliza para crear estados se superposición. Se relaciona con U_3 de la siguiente forma:

$$U_2(\phi,\lambda) = U_3(\pi/2,\phi,\lambda) = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & -e^{i\lambda} \\ e^{i\phi} & e^{i\lambda+i\phi} \end{pmatrix}$$

También podemos incluir la puerta identidad, conocida como I, que no afecta al qubit, pero puede ser útil para introducir ruido. Tiene la siguiente forma:

$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

Relación entre puertas físicas y unitarias

Todas las puertas unitarias que se acaban de describir se pueden escribir como las puertas físicas U_1, U_2 y U_3. A continuación se muestran las relaciones que existen entre estas puertas:

$$X = U_3(\pi,0,\pi) \qquad Y = U_3\left(\pi,\frac{\pi}{2},\frac{\pi}{2}\right) \qquad Z = U_1(\pi)$$

$$H = U_2(0,\pi)$$

$$S = U_1\left(\frac{\pi}{2}\right) \qquad S^t = U_1\left(-\frac{\pi}{2}\right)$$

$$T = U_1\left(\frac{\pi}{4}\right) \qquad T^t = U_1\left(-\frac{\pi}{4}\right)$$

3.2.2 Puertas multiqubit

A diferencia de las puertas unitarias, las puertas multiqubit son aquellas que afectan a dos o más qubits. Este apartado se centrará en las puertas controladas, que serán muy importantes para el algoritmo propuesto.

Una **puerta controlada** está formada por un qubit al que llamaremos *target* y *n* qubits a los que llamaremos *controls*. En resumen, el qubit asignado como *target* cambiará su valor dependiendo de los *controls* que están activos (valen $|1\rangle$).

Para aclarar su funcionamiento es mejor centrarse en una puerta en concreto: **CNOT**. Esta puerta aplica la puerta **X** al qubit correspondiente al *target* solo si el *control* vale $|1\rangle$.

	CNOT
$\|00\rangle$	$\|00\rangle$
$\|01\rangle$	$\|11\rangle$
$\|10\rangle$	$\|10\rangle$
$\|11\rangle$	$\|01\rangle$
MATRIZ	$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$

Tabla 3-4: Resumen de la puerta CNOT, siendo el primer qubit *control* y el segundo *target.*

Se pueden construir estas puertas controladas aplicando al qubit *target* cualquiera de las puertas unitarias que se han descrito en el apartado anterior. La matriz general es la siguiente:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & u_{00} & 0 & u_{01} \\ 0 & 0 & 1 & 0 \\ 0 & u_{10} & 0 & u_{10} \end{pmatrix} \quad \text{siendo } U = \begin{pmatrix} u_{00} & u_{01} \\ u_{10} & u_{10} \end{pmatrix} \quad \text{la matriz de la puerta unitaria.}$$

3.3 Circuitos cuánticos

Un **circuito cuántico** es un conjunto de puertas cuánticas aplicadas a un conjunto de qubits. En la figura 3.X. se puede observar un ejemplo con algunas de las puertas descritas en el apartado 3.2. Los qubits se representan mediante líneas horizontales y las puertas como cajas dibujadas directamente sobre cada línea, estas se aplican sobre los qubits de izquierda a derecha. Las puertas controladas se representan como una caja (*target*) unida con una recta a un punto negro (*control*). Normalmente el estado inicial de cada qubit que compone el circuito se supone $|0\rangle$.

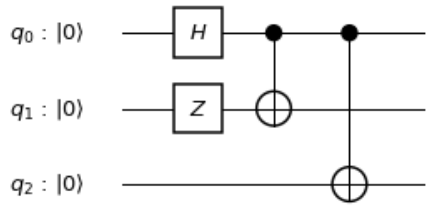


Figura 3-2: Ejemplo de circuito cuántico.

3.4 Algoritmos cuánticos

Un **algoritmo cuántico** es aquel que se ejecuta en un ordenador cuántico. La ventaja de estos algoritmos con respecto a los clásicos es que, al aprovechar los fenómenos de la mecánica cuántica, muchos cálculos se realizan de forma más rápida que en computación clásica.

Existen diversos algoritmos cuánticos, como el algoritmo de Deutsch–Jozsa, el de Shor… pero este Trabajo de Fin de Grado se centrará en el **algoritmo de Grover**.

3.4.1 Algoritmo de Grover

El algoritmo de Grover es un algoritmo cuántico que busca un elemento, o un conjunto de elementos, dentro de una lista desordenada de N elementos. Mientras que con computación clásica esta búsqueda se realiza con una complejidad de O(N), Grover consigue una complejidad de O($\sqrt{N}$) [8].

Se trata de un algoritmo iterativo formado por varios pasos que se describen a continuación.

Superposición uniforme

El primer paso para llevar a cabo el algoritmo es inicializar nuestro circuito con superposición uniforme. Esto se consigue aplicando la puerta H a todos los qubits, que inicialmente se encuentran en el estado $|0\rangle$. Es decir, llevamos a cabo la operación $H^{\otimes n}$, donde n es el número total de elementos de la lista sobre la que estamos realizando la búsqueda.

Oráculo

Este es el paso más complicado del algoritmo. Se trata de encontrar una función que compruebe si un elemento es el buscado. Es decir, hay que encontrar una función **f** tal que f(x) = 0 para todos los elementos **x** de la lista que no buscamos y f(w) = 1, siendo **w** el buscado (**w** puede ser un único elemento o un conjunto).
Para poder aplicar el oráculo es necesario codificar los elementos de la lista de forma que x, w $\in\{0,1\}$. Después definimos una matriz $\mathbf{U_f}$ que se aplicará sobre los elementos de la lista de la siguiente forma:

$$U_f|x\rangle = (-1)^{f(x)}|x\rangle$$

De esta forma el oráculo no modifica los elementos **x**, pero sí cambia el estado del elemento solución, **w**. Este cambio se explicará geométricamente en el apartado 3.4.1.5.

Inversión sobre la media

El objetivo de este paso es incrementar la amplitud del elemento solución. Esto se puede conseguir con la siguiente combinación de puertas:

$$H^{\otimes n}\left(2|0\rangle\langle 0| - I\right)H^{\otimes n}$$

Esta transformación consigue aumentar la amplitud de la solución, **w**. En resumen, con esto se consigue que el valor de la amplitud de **w** aumente el doble de la media del resto de amplitudes, mientras que el valor de la amplitud de los elementos **x** se reduce.

Más en detalle, en esta parte del algoritmo, en un circuito de 3 qubits (lista de 3 elementos), sería necesario aplicar las siguientes puertas:

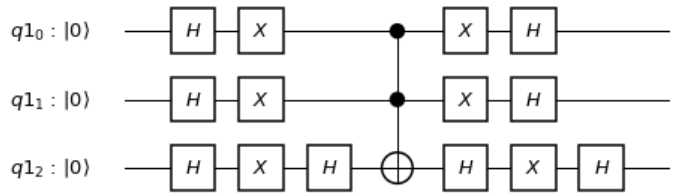


Figura 3-3: Posible implementación de inversión sobre la media para 3 qubits.

Iteraciones

Para que los resultados obtenidos con Grover sean consistentes, es necesario realizar $O(\sqrt{N})$ iteraciones del **oráculo** y la **inversión sobre la media**.

Visualización geométrica

Para comprender mejor cómo funciona el algoritmo de Grover es útil visualizar su funcionamiento geométricamente [9].

En las figuras 3-4 a 3-7 se muestra un ejemplo con 3 qubits, en el que habría 8 posibles soluciones al problema. Cada barra representa la probabilidad de que cada una de estas posibles soluciones realmente lo sea. Como se verá en las figuras, se supone que la solución al problema es la cuarta.

La figura 3-4 muestra el estado inicial del algoritmo de Grover, la superposición uniforme donde todas las posibles soluciones tienen la misma probabilidad.

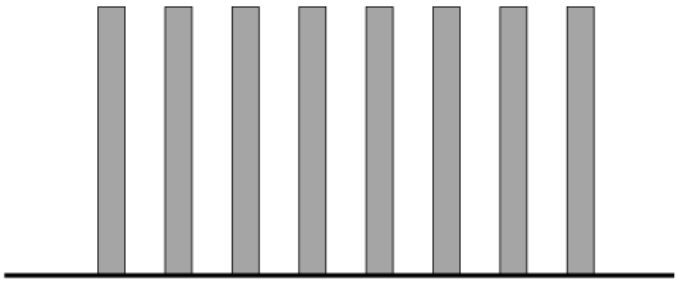


Figura 3-4: Inicialización del algoritmo de Grover con un estado de superposición uniforme.

La figura 3-5 muestra qué ocurriría al aplicar el oráculo, cambiando el estado de la solución (el cuarto).

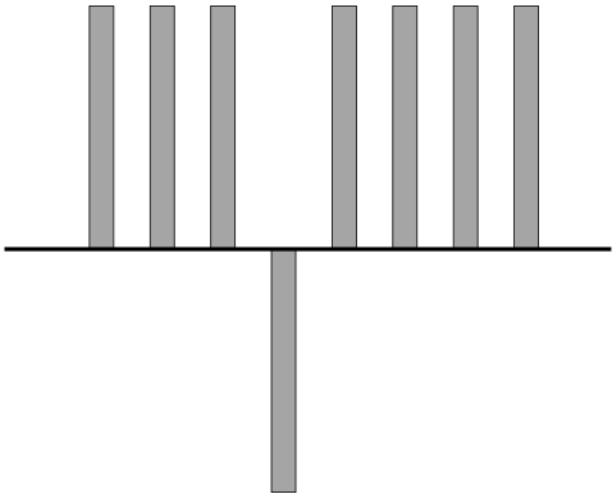


Figura 3-5: Cambio de signo de la solución tras aplicar el oráculo.

La figura 3-6 muestra cómo se calcularía la media para después poder realizar el paso de inversión sobre la media.

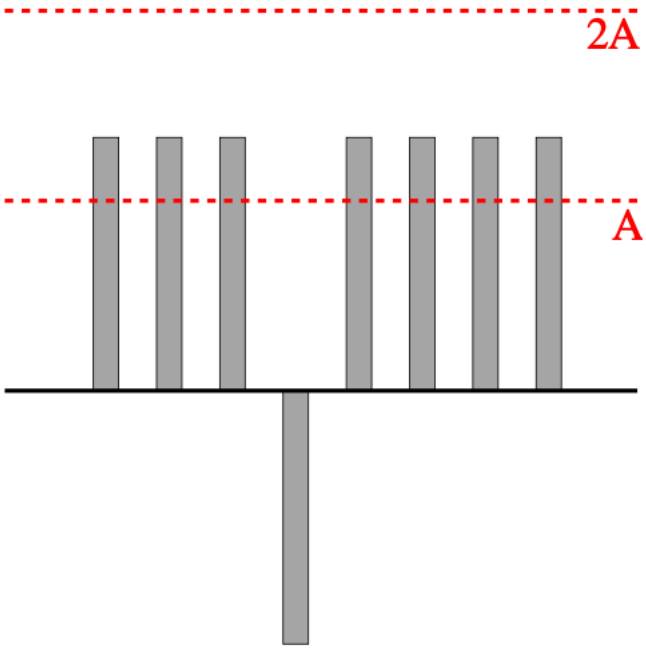


Figura 3-6: Cálculo de la media.

Finalmente, la figura 3-7 muestra cómo quedarían las probabilidades tras la inversión sobre la media y la amplificación de la amplitud. Se puede ver que el estado solución aumenta su amplitud, mientras que los que no lo son la reducen.

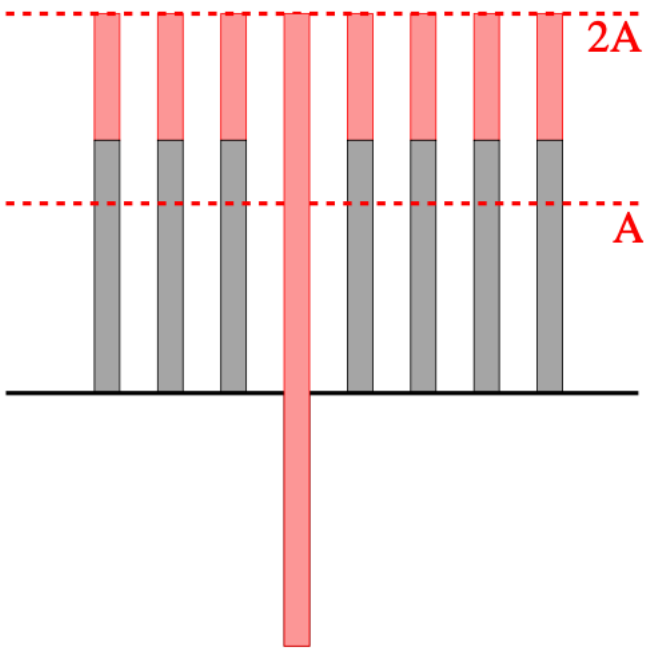


Figura 3-7: Proceso de inversión sobre la media y amplificación de la amplitud.

4 Diseño

En este capítulo se describe el algoritmo realizado durante el desarrollo de este Trabajo de Fin de Grado.

4.1 Problema de satisfacibilidad booleana

El problema de satisfacibilidad booleana (también conocido como SAT) fue el primer problema identificado como perteneciente a la clase de complejidad **NP-completo** por Stephen Cook en el año 1971 [10] [11].

El problema comienza con una lista de variables booleanas x_1, …, x_n. Consideramos un literal a una de las variables x_i (o su negación). Una cláusula es un conjunto de literales. En general, este problema se conoce como N-SAT, donde N es el número de literales por cada cláusula.

Se dice que una fórmula booleana está en forma normal conjuntiva si se trata de una conjunción de *m* cláusulas donde cada cláusula es una disyunción de *n* literales. Es decir, es de la forma:

$$\wedge_{k=1}^{m} C_k, C_k = (x_1 \vee x_2 \vee \ldots \vee x_n)$$

El objetivo es determinar si, dada una fórmula en forma normal conjuntiva, existe alguna combinación de valores para sus literales que la satisfaga.

Este problema se puede entender mejor con un ejemplo. Consideremos la siguiente fórmula en forma normal conjuntiva: $(x_1 \vee x_3) \wedge (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_3)$, donde $\neg$ denota la negación del literal. Es fácil ver que en este caso si elegimos los valores $(x_1, x_2, x_3) = (V, V, V)$ se satisface la fórmula. Observamos que hay 2^n posibles conjuntos de valores que pueden resolver el problema, por lo que se trata de un problema de alto coste computacional. El número de posibles soluciones crece exponencialmente con el número de literales

4.1.1 Problema ‘Exactly-1 3-SAT’

El problema ‘Exactly-1 3-SAT’ es un caso especial del problema N-SAT. En este caso el número de literales por cláusula es exactamente 3. Este problema consiste en encontrar aquellas expresiones que satisfacen la fórmula conteniendo exactamente un literal verdadero.

Tomemos como ejemplo la siguiente fórmula en forma normal conjuntiva:

$$(x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3)$$

Haciendo las pertinentes comprobaciones se puede comprobar que la solución sería $(x_1, x_2, x_3) = (V, F, V)$, la cual hace que solo haya un literal verdadero por cláusula, satisfaciéndose así la fórmula. En el resto del documento, en lugar de utilizar V o F, se utilizará **0** para denotar **falso** y **1** para denotar **verdadero**.

4.2 Descripción del algoritmo

Para resolver el problema de satisfacibilidad booleana se ha desarrollado un algoritmo cuántico mediante un circuito que utiliza las puertas cuánticas descritas en el apartado 3.2.

4.2.1 Introducción al algoritmo

El objetivo inicial de este Trabajo de Fin de Grado era la implementación del algoritmo para resolver el problema de satisfacibilidad booleana para cualquier número de literales y cláusulas, pero debido a diversas limitaciones hardware que se explican en el apartado 5.3.1 tuvo que limitarse a tres cláusulas y tres literales como máximo. Más adelante también se explicará que este problema es fácilmente escalable, pero que depende íntegramente de las características del ordenador cuántico.

Como se explicó en el apartado 2.2, este trabajo surgió a partir de un error detectado en un artículo que intentaba resolver el problema 'Exactly-1 3-SAT'. El algoritmo implementado se basa en el algoritmo de Grover, es decir, se realiza una búsqueda dentro de todas las posibles soluciones hasta encontrar la correcta (o las correctas).

4.2.2 Primera aproximación

En primer lugar, se plantearon varias posibles soluciones para corregir el error que cometía el algoritmo inicial. Ambas consistían en modificar la parte del oráculo del algoritmo de Grover, introduciendo nuevas puertas cuánticas para que se cumpliera la condición de *exactly-1*, es decir, para que en la solución solo hubiera un literal verdadero por cada cláusula.

El número de qubits utilizados es muy importante, ya que actualmente supone una limitación al no disponer de ordenadores cuánticos con más de 16 qubits a los que pueda acceder cualquier usuario. Por ello, se propusieron dos soluciones: una de ellas minimizaba el número de qubits, aumentando el número de puertas cuánticas, y la otra hacía justo lo contrario. En el próximo apartado se describirán con más detalle estas dos soluciones. También se elaboró una tercera solución que utiliza puertas que aún no están disponibles en Qiskit

4.3 Diseño del algoritmo

En este apartado se describe con todo detalle el diseño del algoritmo, definiendo las partes del circuito cuántico. Durante el desarrollo de este Trabajo de Fin de Grado se han implementado varias soluciones equivalentes, pero todas ellas tienen una parte en común que se describirá a continuación.

4.3.1 Parte común

Las tres implementaciones que se describen en los siguientes apartados tienen una parte en común: el algoritmo de Grover. Los pasos de los que consta este algoritmo se explicaron con detalle en el apartado 3.4.1. En la figura 4-1 se puede ver un esquema de la parte común del circuito de todas las implementaciones. En los siguientes apartados se mostrará un ejemplo del circuito de cada una de las partes

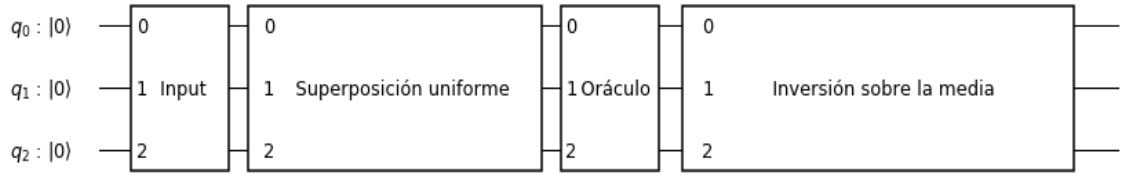


Figura 4-1: Esquema de la parte común de las implementaciones.

En las tres implementaciones que se van a detallar más adelante la única parte que varía es la del **oráculo**, el resto se mantiene prácticamente sin ningún cambio. Otra de las partes del algoritmo que es importante es la de la introducción de datos, señalada en la figura 4-1 como **input**, que se explicará en el apartado 4.3.1.1.

4.3.1.1 Introducción de los datos

El algoritmo comienza introduciendo los datos en el circuito. Esta es una de las partes que más limita, ya que es necesario codificar los datos de forma binaria. Al tratarse de un problema cuya entrada es un conjunto de variables booleanas, se necesita un qubit por cada variable. Cabe destacar que en el caso en el que la entrada no fuese una variable booleana, que tuviese, por ejemplo, tres posibles valores, el número de qubits necesarios se duplicaría, ya que se necesitarían dos por cada variable.

En la figura 4-2 se muestra un ejemplo de un circuito en el que la entrada es la siguiente cláusula: $(\neg x_1 \vee x_2 \vee \neg x_3)$. Como se explicó anteriormente, el estado inicial de los qubits es $|0\rangle$. Se utilizan las puertas unitarias X para cambiar el estado a $|1\rangle$, que representa un literal negado.

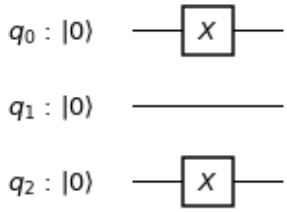


Figura 4-2: Ejemplo de introducción de datos en el circuito.

4.3.1.2 Estado inicial

El algoritmo parte de un estado inicial de superposición uniforme. Para conseguir esto basta con aplicar la puerta de Hadamard a todos los qubits de la entrada. La figura 4-3 muestra cómo se iría completando el circuito tras la superposición uniforme.

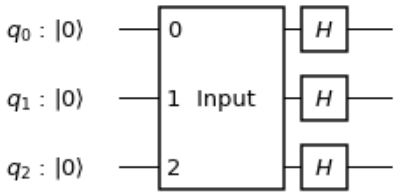


Figura 4-3: Creación de superposición uniforme.

4.3.1.3 Inversión sobre la media

Esta parte del algoritmo puede ser implementada de varias formas. En la figura 3-3 se mostró una de las posibilidades y en la figura 4-4 se puede ver otra equivalente. En el último caso se utiliza una puerta z controlada.

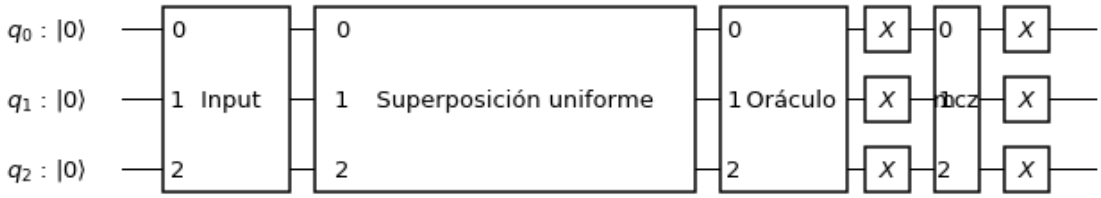


Figura 4-4: Posible implementación de inversión sobre la media en la que se utiliza una puerta Z controlada donde los dos primeros qubits son los controles y el tercero es el target.

4.3.1.4 Número de iteraciones

Como se explicó en el apartado 3.4.1.4, el número de iteraciones que se necesitan para obtener un resultado coherente es $O(\sqrt{N})$. Como en este caso el número de literales por cláusula es 3 (N = 3), se necesitan hacer **2 iteraciones** tanto del oráculo como de la parte de la inversión sobre la media.

4.3.1.5 Mediciones

La parte final del algoritmo consiste en medir y obtener las probabilidades. Para ello, en Qiskit se crea un tipo de registro utilizado para medir el valor de cada uno de los qubits, que puede ser $|0\rangle$ o $|1\rangle$. En la figura 4-5 se puede ver cómo sería el circuito.

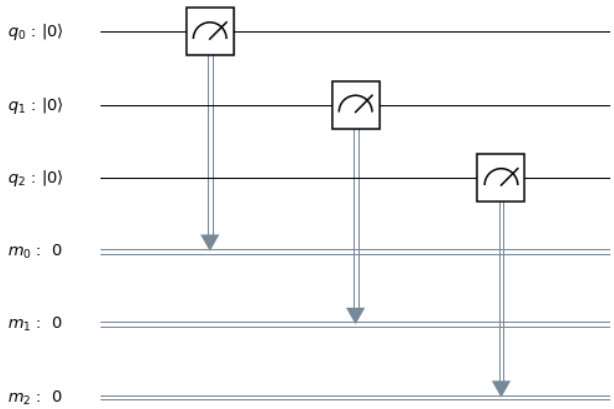


Figura 4-5: Circuito con registros para medir el valor de los qubits.

Como ya se explicó, al medir un qubit este colapsa y pierde la propiedad de superposición, por lo que solo puede valer $|0\rangle$ o $|1\rangle$. Entonces, en cada una de las mediciones se obtendrá un único valor para cada una de las variables. Al tratarse de un algoritmo probabilístico es necesario ejecutarlo muchas veces para obtener resultados coherentes. Al simulador se le puede indicar el número de *shots* que queremos realizar. Un *shot* es una ejecución del algoritmo. En todas las pruebas llevadas a cabo se han realizado 1000 *shots*. Así se obtiene la probabilidad de que la solución sea los valores 'XXX'. Por ejemplo, en la figura 4-6 se muestra el resultado de una ejecución donde la solución es $(x_1, x_2, x_3) = (0,0,0)$.

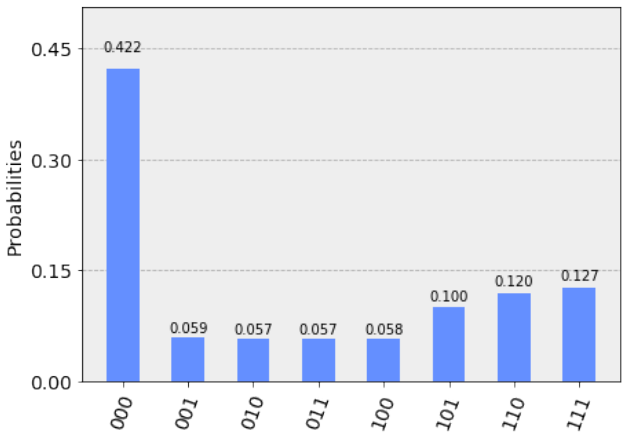


Figura 4-6: Histograma que muestra un ejemplo de ejecución del algoritmo donde la solución es $(x_1, x_2, x_3) = (0,0,0)$.

4.3.2 Primera implementación

El algoritmo comienza con un estado de superposición uniforme, es decir, uno en el que obtendríamos cada posible solución con la misma probabilidad. Todas las implementaciones que se van a describir en este y en los próximos apartados tienen la misma finalidad: modificar estas probabilidades para conseguir aumentar la de la combinación correcta (o combinaciones) y reducir las del resto.

Para esta primera implementación se realizó un circuito con 9 qubits, 3 se utilizaban para la entrada y los 6 restantes eran auxiliares. Tres de estos qubits auxiliares tienen inicialmente el valor $|0\rangle$. A cada cláusula le corresponde uno de los qubits auxiliares. Construimos un circuito para cada una de las cláusulas, de forma que se aplica una puerta X al qubit correspondiente si y solo si la cláusula tiene exactamente un literal verdadero. Con esto se consigue que los qubits auxiliares de cada cláusula valgan $|1\rangle$ si estas satisfacen la condición. Finalmente habría que comprobar que los tres qubits valieran $|1\rangle$ y así se obtendría la solución.

En resumen, la idea es la siguiente: se aplica la puerta X por cada literal que aparezca negado y, usando puertas CNOT, se cambia el qubit auxiliar de dicha cláusula a $|1\rangle$ si se satisface que haya exactamente un literal verdadero. Para comprobar esto último se hacen dos comprobaciones: primero se ve que el número de literales verdaderos sea impar, y después se descarta la solución que contiene 3 verdaderos, quedándose así como correcta la que solo tiene 1.

La parte más complicada es la de las puertas CNOT. Cuando se desarrolló esta implementación solo estaban disponible en Qiskit las puertas controladas para un target y dos controles. Pero en este caso se necesitan puertas con tres controles, por lo que se hicieron 'manualmente'. En la figura 4-7 se puede ver un ejemplo de una puerta triple controlada (CCCNOT) construida a base de puertas dobles controladas (CCNOT). Los tres primeros qubits son los controles, el cuarto es el target y el quinto un auxiliar.

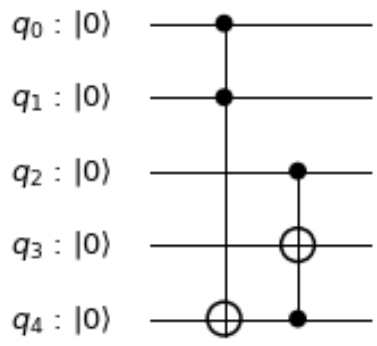


Figura 4-7: Posible implementación de una puerta triple controlada (CCCNOT).

Uno de los problemas que surge con esta implementación es que, al dividir el circuito en tres partes y no poder medir los qubits hasta el final, para no perder la superposición cuántica, tras cada subcircuito habría que volver al estado inicial. Esto solo se puede conseguir aplicando a los qubits las mismas puertas de forma inversa. Es decir, observando la figura 4-8, para devolver al último qubit al estado que tenía antes de aplicarle las puertas, tenemos

que volver a aplicarlas. Claramente eso provoca que el número de puertas necesarias sea altísimo.

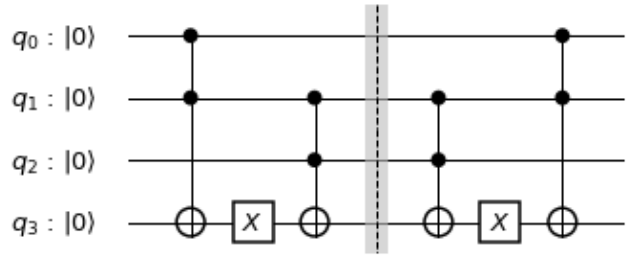


Figura 4-8: Ejemplo de devolver al último qubit a su estado inicial.

Con esta solución se obtenían resultados correctos, pero solo en el simulador. En los ordenadores cuánticos reales disponibles, debido al gran número de puertas que se utilizaban y al tiempo de decoherencia, se obtenían resultados aleatorios. Esto se explicará con más detalle en el apartado 5.2.1.

Por motivos de espacio, en el anexo A se incluirá el circuito completo que surge de esta implementación.

4.3.3 Segunda implementación

Esta implementación pretende mejorar la anterior. Como se explicó en el apartado anterior, debido al alto número de puertas utilizadas en la primera implementación se obtenían resultados aleatorios el un ordenador cuántico real. En este caso se reduce notablemente el número de puertas incrementando la cantidad de qubits auxiliares.

Para llevar a cabo esta implementación se utilizan 14 qubits, que es el máximo disponible en los ordenadores de IBM Quantum Experience. De esta forma se logran conseguir resultados coherentes no solo en el simulador, sino también en un ordenador cuántico real, exactamente en el de Melbourne, de 14 qubits.

La idea es muy parecida a la del apartado anterior. La única diferencia es que, al haber 5 qubits auxiliares más, no es necesario revertir algunos qubits a un estado anterior. Esto reduce notablemente el número de puertas necesarias. El circuito completo se encuentra en el anexo B.

4.3.4 Tercera implementación

Esta última implementación se desarrolló tras obtener un código que aún no está disponible en Qiskit. Gracias a esto se pudieron utilizar puertas multicontrol de todos los controles que se necesitaran. En las anteriores implementaciones se tenía que desarrollar manualmente la puerta CCCNOT a partir de dos CCNOT, pero ahora eso ya no es necesario. Por estos motivos el número de qubit utilizados en esta implementación es 7.

Este recurso reduce más aún el número de puertas necesarias, pero tiene una gran limitación: no se puede ejecutar en un ordenador cuántico real porque no está implementado. Por tanto, obtenemos una solución mucho más simple y cómoda que las anteriores pero que

solo se puede probar en el simulador. Se espera que, en el futuro, cuando se incluyan estas puertas en Qiskit, esta solución se pueda probar en ordenadores cuánticos reales.

Por motivos de espacio, el circuito completo se puede visualizar en el anexo C.

4.4 Resumen de implementaciones

La tabla 4-2 contiene un resumen de las características principales de cada una de las implementaciones descritas en el apartado anterior. En ella se especifica, para cada una de las implementaciones, si se han ejecutado en el simulador y/o en el ordenador cuántico real, el número de qubits utilizado y el número medio de puertas, divididas por tipo y en total.

	PRIMERA	**SEGUNDA**	**TERCERA**
Simulador	Sí	Sí	Sí
Ordenador cuántico	No	Sí	No
Número de qubits	9	14	7
Número medio de puertas X	68	49	48
Número medio de puertas H	20	16	16
Número medio de puertas controladas	128	90	16
Número medio de puertas totales	216	155	80

Tabla 4-1: Resumen de las tres implementaciones propuestas

Observando la tabla se puede ver que, con mucha diferencia, el número de puertas más utilizadas son las de control. Por esta razón la tercera implementación es la que tiene el número medio de puertas totales menor. Esto se debe a que, en las otras dos implementaciones, al no estar aún en Qiskit las puertas triples controladas (CCCNOT),

tenían que hacerse ‘a mano’. Hacer las puertas triples controladas a partir de otras necesita tres puertas dobles controladas, lo que explica la diferencia en cuando al número utilizado.

Cuando se realizó la tercera implementación, gracias a un código cedido por el equipo de IBM Research, se disponía de estas puertas triples controladas en Qiskit. Aparte de utilizar este código para la puerta CCCNOT, también se aprovechó para hacer la puerta CCCZ en la parte de inversión sobre la media.

5 Pruebas y resultados

En este capítulo se mostrarán los resultados obtenidos con las diferentes implementaciones, tanto en el simulador como en ordenadores cuánticos reales.

5.1 Limitaciones del algoritmo de Grover

En primer lugar, es importante explicar las limitaciones que presenta el algoritmo de Grover. Como ya se explicó en el capítulo 3, este algoritmo realiza una búsqueda en una lista desordenada amplificando la amplitud de la solución y minimizando la de las que no son solución.

El problema surge cuando hay más de ciertas soluciones. Llamando n al número de elementos de la lista sobre la que se hace la búsqueda, si el número de soluciones es igual o mayor que $n/2 - 1$, tras aplicar el algoritmo se obtendrá el resultado opuesto al esperado. Es decir, el algoritmo resaltará como soluciones válidas a aquellas que no lo son y como no válidas a las que sí lo son.

Esto aplicado al problema que hemos tratado, donde $n = 2^3 = 8$, indica que si tomamos una fórmula en forma normal conjuntiva que tengas más de 3 posibles soluciones obtendremos como soluciones justo las que no lo son. Por ejemplo, si tomamos la fórmula que tiene una única cláusula (x_1, x_2, x_3), se ve claramente que hay tres posibles soluciones: (1,0,0), (0,1,0) y (0,0,1). En la figura 5-1 se puede ver el resultado que se obtiene en cualquiera de las tres implementaciones descritas en el apartado 4.3.

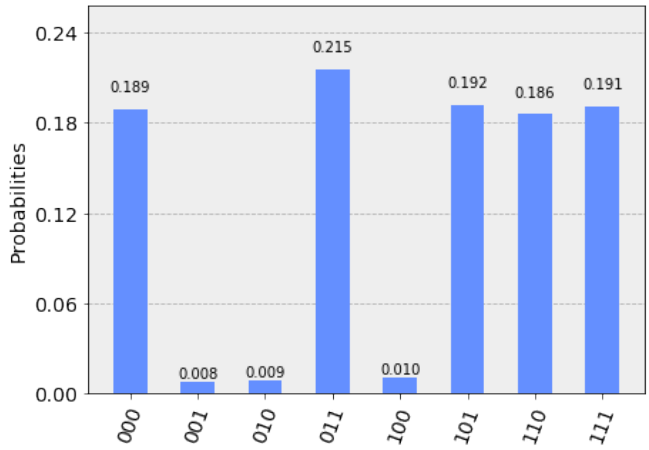


Figura 5-1: Ejemplo de histograma en el que el algoritmo de Grover resalta las combinaciones que no son soluciones.

5.2 Pruebas en el simulador

Para poder comprobar el correcto funcionamiento de las tres posibles implementaciones del algoritmo desarrollado se ha utilizado el simulador que ofrece IBM.

En primer lugar, se elaboró un programa en Python que resolvía este problema de forma clásica. Se calcularon todas las posibles permutaciones de tres cláusulas, cada una con tres literales, obteniéndose que hay 48 fórmulas con una única solución y 72 con dos posibles soluciones. Es fácil comprobar que las fórmulas que tienen dos posibles soluciones son aquellas que están formadas solo por dos cláusulas, o por tres, pero teniendo dos de ellas repetidas. Por el problema explicado en el apartado 5.1. no se han calculado las que tienen 3 soluciones porque no se podría probar su funcionamiento, se obtendrían resultados erróneos.

Para realizar las pruebas se ha realizado un programa en Python usando Qiskit que recibe como entrada una fórmula en forma normal conjuntiva y crea el circuito para resolver el problema. Finalmente muestra la solución del problema en forma de histograma, donde las soluciones son aquellas cuyos valores de probabilidad son más altos. En el histograma se muestra falso como 0 y verdadero como 1, como ya se explicó anteriormente.

El simulador ofrecido por IBM, hoy en día, cuenta con un máximo de 32 qubits, pero cuando se comenzaron a realizar estas pruebas solo disponía de 20. Con este dispositivo se simula un comportamiento totalmente estable y con conexiones entre todos los qubits.

En la figura 5-2 se puede ver el ejemplo del resultado obtenido cuando la entrada es la fórmula , que tiene una única solución válida. También se muestra el resultado obtenido cuando la entrada es $(\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3)$ en la figura 5-3, en este caso hay dos soluciones válidas.

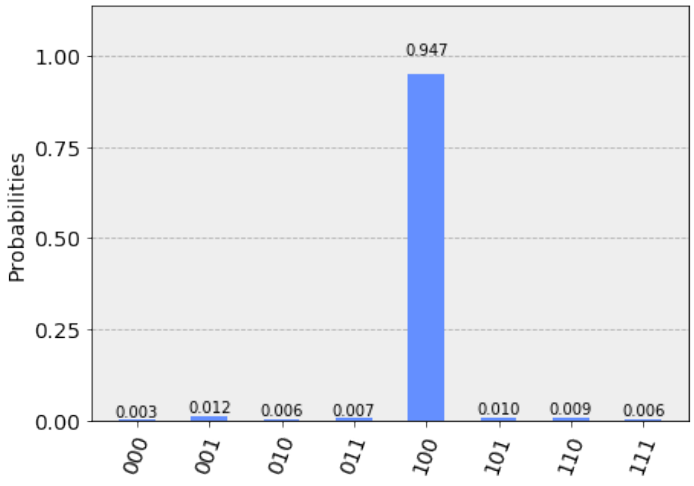


Figura 5-2: Histograma que muestra un ejemplo de ejecución del algoritmo en el simulador donde la solución es $(x_1, x_2, x_3) = (1,1,0)$.

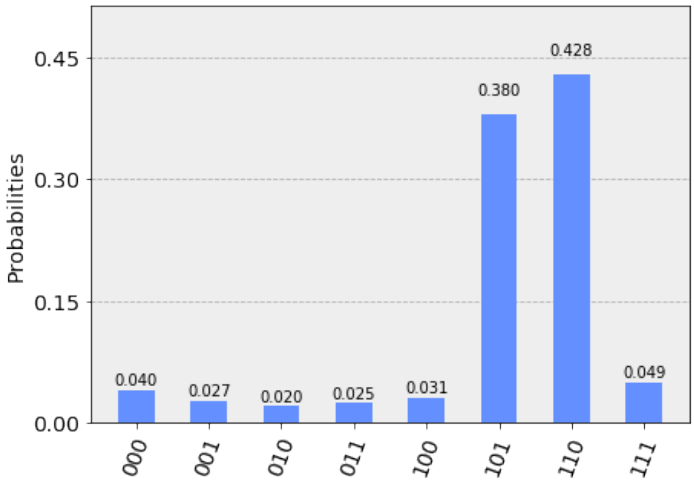


Figura 5-3: Histograma que muestra un ejemplo de ejecución del algoritmo en el simulador donde la solución es $(x_1, x_2, x_3) = (1,1,0)$ y $(x_1, x_2, x_3) = (1,0,1)$

5.3 Pruebas en ordenadores cuánticos reales

A la hora de realizar pruebas en los ordenadores cuánticos reales que ofrece IBM surgen tren problemas: el número de qubits, el tiempo de decoherencia y la topología de estos ordenadores.

Dos de ellos tienen 5 qubits, los cuales se descartan porque no son suficientes para poder probar el algoritmo. Por tanto, solo queda el ordenador de 16 qubits, ubicado en Melbourne. La topología de estos tres ordenadores será detallada en el apartado 5.3.1.

Comparados con los obtenidos con el simulador, los resultados de la ejecución del algoritmo en el ordenador cuántico de 16 qubits son mucho más irregulares. En la figura 5-4 se puede ver un ejemplo de un resultado obtenido con el ordenador cuántico de Melbourne cuya entrada es: $(x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3)$. Comparado con el resultado de la figura 5-2, este último es mucho más estable, ya que fue realizado en un simulador.

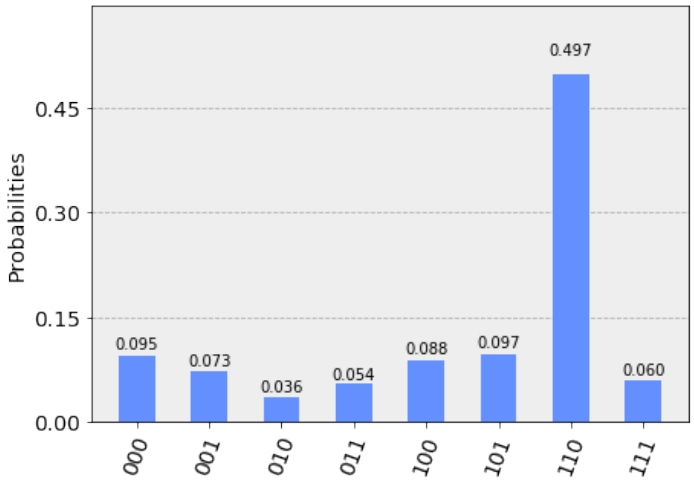


Figura 5-4: Histograma que muestra un ejemplo de ejecución del algoritmo en el simulador donde la solución es $(x_1, x_2, x_3) = (1,1,0)$.

Como ya se explicó en el apartado 4.4, la única implementación de las realizadas que puede ser probada en un ordenador cuántico real es la segunda. La tercera implementación no se puede ejecutar en un ordenador real ya que la puerta triple controlada no está implementada aun en estos. Y la segunda implementación no puede probarse debido al gran número de puertas que requiere y al tiempo de decoherencia cuántico, que será explicado en el siguiente apartado.

5.3.1 Decoherencia cuántica

La decoherencia cuántica es el proceso que produce la pérdida de coherencia de un estado cuántico. Puede verse como la pérdida de información de un sistema debido a la interacción con el entorno.

La decoherencia constituye uno de los mayores obstáculos al que debe enfrentarse la realización práctica de los ordenadores cuánticos, ya que estos esperan que los estados coherentes se conserven para poder realizar realmente un cálculo cuántico sin errores [12].

Este concepto aplicado en este Trabajo de Fin de Grado supone que, si el algoritmo no se ejecuta en menos de cierto tiempo, los resultados obtenidos serán aleatorios, ya que se perderá la estabilidad del ordenador cuántico y todo lo que se mida después de ese tiempo no será válido.

Por este motivo la primera implementación, que al contener más puertas tardaba más tiempo, no se puede ejecutar en un ordenador cuántico real. Hoy en día el tiempo de decoherencia conseguido por las empresas que desarrollan ordenadores cuánticos aún es muy bajo, lo que no permite realizar cálculos complejos correctos. Las características de los ordenadores de los que dispone IBM, que son los utilizados en este Trabajo de Fin de Grado, están detalladas en las tablas 5-1, 5-2 y 5-3, además de los errores de las puertas y de cada qubit individualmente. Se puede ver que el tiempo de decoherencia de estos ordenadores es muy bajo, lo cual provoca que las pruebas no se puedan realizar en ellos. Esto explica por qué se han realizado la gran mayoría en el simulador.

La relajación de la energía es un proceso de decoherencia donde el estado excitado $|1\rangle$ cae al estado $|0\rangle$. El tiempo que lleva este proceso se denomina T_1. El desfase es otro proceso de decoherencia y, a diferencia de la relajación energética, afecta solo a los estados de superposición. Puede entenderse únicamente en una configuración cuántica, ya que no tiene un análogo clásico. La constante de tiempo T_2 hace referencia a este efecto de desfase. Ambos valores aparecen también en las tablas.

qubit	multi_qb_gate_error	T1 (µs)	T2 (µs)	readout_error	gate_error
Q0		81.865242257752	22.1022475507186	0.0242	0.00389717312799298
Q1	CX1_0: 0.04 CX1_2: 0.1	65.727183122093	110.136650944182	0.0384999999999999	0.01097021553278709
Q2	CX2_3: 0.06,	78.497285914424	153.953556231687	0.0445999999999999	0.00915398983624171
Q3		63.818590660632	59.6198155549304	0.3155	0.00553505383866059
Q4	CX4_3: 0.04, CX4_10: 0.03	46.788558934394	33.1541138396252	0.0592000000000000	0.00320275467886421
Q5	CX5_4: 0.06, CX5_6: 0.06 CX5_9: 0.09	27.002471422731	50.9764203642329	0.0504	0.00598458865092166
Q6	CX6_8: 0.03,	83.835137687972	75.6475747730301	0.0504	0.00256758000370682
Q7	CX7_8: 0.04,	38.610458308152	65.3138212129265	0.1977999999999999	0.00315668587625328
Q8		97.599313245746	197.564064105082	0.0630999999999999	0.00272234054035758
Q9	CX9_8: 0.06, CX9_10: 0.05	37.803211232334	57.8686567417568	0.0321000000000000	0.00512155266917557
Q10		48.2450753867945	58.1410570260149	0.0508999999999999	0.00525516612438381
Q11	CX11_3: 0.12, CX11_10: 0.1 CX11_12: 0.14	62.0721150101077	102.034091671966	0.3349	0.05582021354697997
Q12	CX12_2: 0.12,	79.1648217988021	112.392260088729	0.128299999999999	0.00928288204005833
Q13	CX13_1: 0.15, CX13_12: 0.04	24.2681675019317	53.2730400256833	0.0373	0.012085419563190647

Tabla 5-1: Características del ordenador cuántico de Melbourne.

qubit	multi_qb_gate_error	T1 (µs)	T2 (µs)	readout_error	gate_error
Q0		55.6547298922308	18.5404354867565	0.042750000000000	0.00154526145869071
Q1	CX1_0: 0.03	44.4138160339497	8.586604549207443	0.0615	0.00429336803764668
Q2	CX2_0: 0.03, CX2_1: 0.07	35.3637042557895	47.68219611384455	0.1887499999999999	0.00463696929978452
Q3	CX3_2: 0.07, CX3_4: 0.08	38.3360207745528	41.6273377836503	0.3602499999999999	0.00326268159384890
Q4	CX4_2: 0.08	45.395634975394	6.92752039410608	0.296000000000000	0.00927750563003015

Tabla 5-2: Características del ordenador cuántico de Tenerife

qubit	multi_qb_gate_error	T1 (µs)	T2 (µs)	readout_error	gate_error
Q0	CX0_1: 0.06 CX0_2: 0.06	67.683512531725	38.2935210160145	0.0582499999999999	0.00979334192841940
Q1	CX1_2: 0.06,	81.019257435795	53.576725156926	0.2604999999999999	0.00309091762057711
Q2		87.288367322450	62.7463577006647	0.0124999999999999	0.00927750563003015
Q3	CX3_2: 0.08, CX3_4: 0.04	59.1749838568902	24.20932729755998	0.1089999999999999	0.00652712633346241
Q4	CX4_2: 0.04	51.754843975781	32.5041725080992	0.3800	0.00429336803764668

Tabla 5-3: Características del ordenador cuántico de Yorktown.

En las figuras 5-1, 5-2 y 5-3 se muestran diagramas con la topología de los ordenadores cuánticos de los que dispone IBM para uso público. Están localizados en Melbourne, Tenerife y Yorktown, y tienen, respetivamente, 14, 5 y 5 qubits. Cada círculo representa un qubit y las flechas que hay entre ellos las conexiones. Por ejemplo, en el ordenador de Melbourne no se podría aplicar directamente una puerta CCNOT entre los qubits 1 y 12. Para hacerlo habría que hacer un swap entre, por ejemplo, el 2 y el 1 para que así estuviera conectado con el 12, y posteriormente volver a hacer un swap para devolver el valor del qubit 1 a su sitio [13].

La relajación de la energía es un proceso de decoherencia donde el estado excitado $|1\rangle$ cae al estado $|0\rangle$. El tiempo que lleva este proceso se denomina T_1. El desfase es otro proceso de decoherencia y, a diferencia de la relajación energética, afecta solo a los estados de superposición. Puede entenderse únicamente en una configuración cuántica, ya que no tiene un análogo clásico. La constante de tiempo T_2 hace referencia a este efecto de desfase. Ambos valores aparecen también en las tablas.

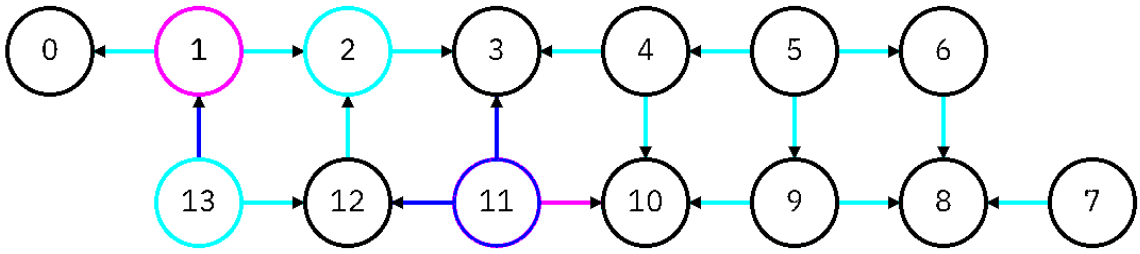


Figura 5-5: Topología del ordenador cuántico de Melbourne.

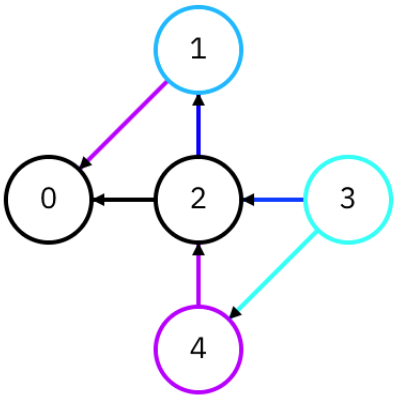


Figura 5-6: Topología del ordenador cuántico de Tenerife.

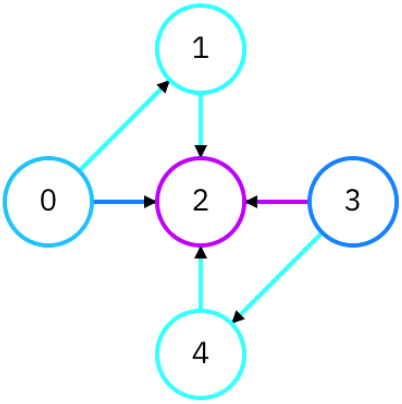


Figura 5-7: Topología del ordenador cuántico de Yorktown.

6 Conclusiones y trabajo futuro

6.1 Conclusiones

Una de las partes más importantes de este Trabajo de Fin de Grado ha sido el estudio y comprensión del funcionamiento y los principios básicos de la computación cuántica. Desde el análisis de las diferencias respecto a la computación clásica, hasta las leyes físicas que rigen su funcionamiento. Este proceso de investigación ha sido la base para después poder desarrollar el algoritmo que resuelve el problema de satisfacibilidad booleana.

La computación cuántica es una rama de la computación que está iniciándose aun, además de estar en continuo cambio y desarrollo. Actualmente se trata de un ámbito con muchas limitaciones, desde el tiempo de decoherencia hasta el número de qubits y el problema de la introducción de los datos.

La mayoría de los algoritmos desarrollados son teóricos, ya que es difícil llevarlos a la práctica por las limitaciones de los ordenadores cuánticos reales disponibles. Estas limitaciones cesarán en el momento en el que el hardware mejore.

El algoritmo desarrollado en este Trabajo de Fin de Grado es una forma de unir todos los conocimientos adquiridos durante el proceso de investigación. Se trata de una implementación mejorable y con limitaciones, pero que, cuando los ordenadores disponibles mejoren sus características, será totalmente escalable a cualquier número de literales.

6.2 Trabajo futuro

En este apartado se proponen distintas formas de mejorar y ampliar el funcionamiento del algoritmo desarrollado en este Trabajo de Fin de Grado.

En primer lugar, se podría ampliar el de literales por cada cláusula hasta llegar al caso general, N-SAT, donde N sería un número de literales cualquiera. Actualmente, con el número de qubits que tienen los ordenadores cuánticos disponibles esto no es posible, debido a la cantidad de qubits auxiliares necesarios. Cuando las características de estos ordenadores mejoren, esta mejora podría llevarse a cabo.

También se podría ampliar el número de cláusulas que contiene la fórmula en forma normal conjuntiva. En este caso sucede lo mismo que con el número de literales. Al necesitarse tantos qubits auxiliares no se dispone de un ordenador cuántico con las características adecuadas para llevarlo a cabo.

En tercer lugar, se podría implementar un algoritmo que en lugar de resolver el problema 'Exactly-1 3-SAT', resolviera el problema sin esta restricción, es decir, N-SAT. Esto supone un problema porque en este caso hay más soluciones posibles, y si se usara el algoritmo de Grover no se obtendrían resultados válidos. Como ya se explicó, Grover funciona bien cuando el número de posibles soluciones es el menor posible. Por tanto, habría que diseñar un algoritmo totalmente diferente que no utilizase Grover para poder resolver este problema. Esto último también sería útil para resolver el caso que se describe en el apartado 5.1, en el que se obtienen como soluciones del problema justo las que no lo son.

Finalmente, otro punto de mejora serían los resultados obtenidos en un ordenador cuántico real. De las tres implementaciones desarrolladas, solo una de ellas puede funcionar en un ordenador cuántico real y sus resultados no son tan buenos como en el simulador, claramente. Esto depende principalmente de las características del ordenador cuántico sobre el que se lancen las pruebas, especialmente del tiempo de decoherencia. Por lo que es un ámbito más físico, que se escapa de los conocimientos informáticos.

7 Bibliografía

[1] E. Juan Pablo Rúa Vargas y J. W. B. B., «Avances en Sistemas e Informática,» vol. 6, Septiembre de 2009.

[2] T. M. Braje y J. A. Cortese, «Loading Classical Data into a Quantum Computer,» *Massachusetts Institute of Technology,* March 2018.

[3] G. Nannicini, «An Introduction to Quantum Computing, Without the Physics,» November 20, 2018.

[4] V. Autores, «Quantum Information Science Kit,» [En línea]. Available: https://qiskit.org. [Último acceso: Junio 2019].

[5] V. Autores, «Slack Community,» [En línea]. Available: https://qiskit.slack.com. [Último acceso: Junio 2019].

[6] M. Nakahara y T. Ohmi, Quantum Computing: From Linear Algebra to Physical Realizations, CRC Press, 2008.

[7] V. Autores, «IBM Quantum Experience,» [En línea]. Available: https://quantumexperience.ng.bluemix.net/. [Último acceso: Abril 2019].

[8] I. Chuang y M. Nielsen, Quantum Computation and Quantum Information, Cambridge University Press, 2000.

[9] O. Morsch, Quantum Bits and Quantum Secrets: How Quantum Physics is revolutionizing Codes and Computers, Wiley-VCH, 2008.

[10] H. S. Wilf, «Algorithms and Complexity,» 1994.

[11] M. Garey y D. S. Johnson, Computers and Intractability: A Guide to the Theory of NP-completeness, W. H. Freeman and Company, 1979.

[12] M. A. Schlosshauer, Decoherence and the Quantum-To-Classical Transition, Springer, 2010.

[13] V. Autores, «Quantum Computing IBM,» [En línea]. Available: https://quantum-computing.ibm.com. [Último acceso: Junio 2019].

[14] G. Benenti, G. Casati y G. Strini, Principles od Quantum Computation and Information, World Scientific, 2004.

[15] S. Willi-Hans y Y. Hardy, Problems and Solutions in Quantum Computing and Quantum Information, World Scientific, 2018.

[16] J. Jones y D. Jaksch, Quantum Information, Computation and Communication, Cambridge University Press, 2012.

[17] M. Hirvensalo, Quantum Computing, Springer, 2001.

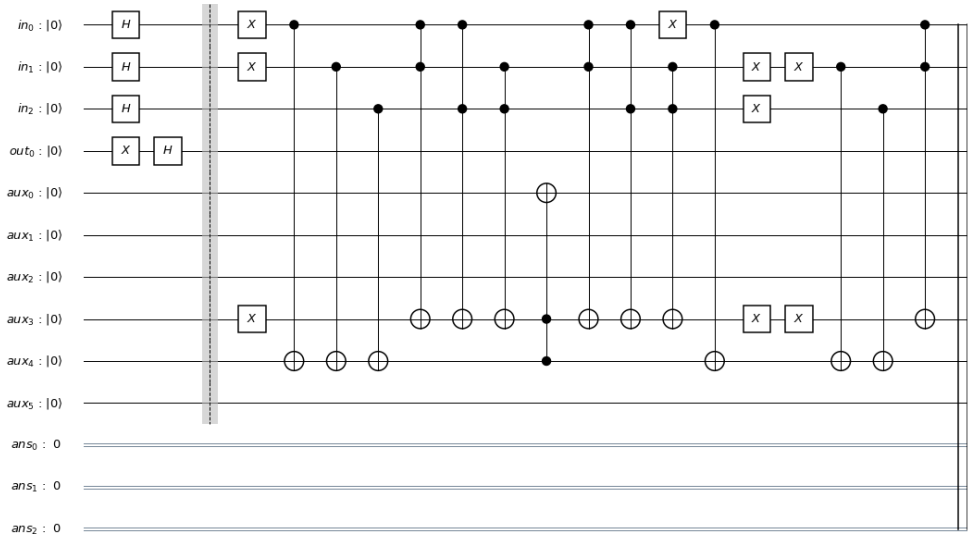
Anexos

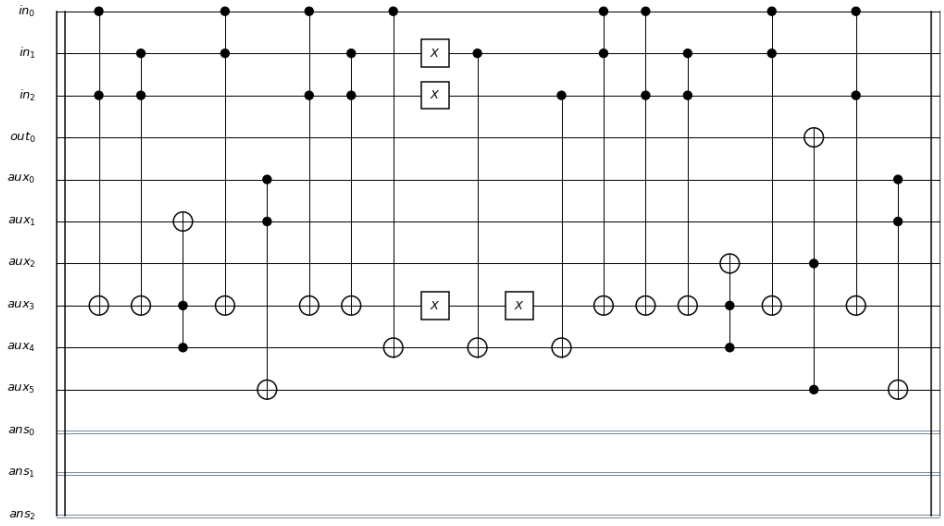
A Ejemplo de circuito de la primera implementación

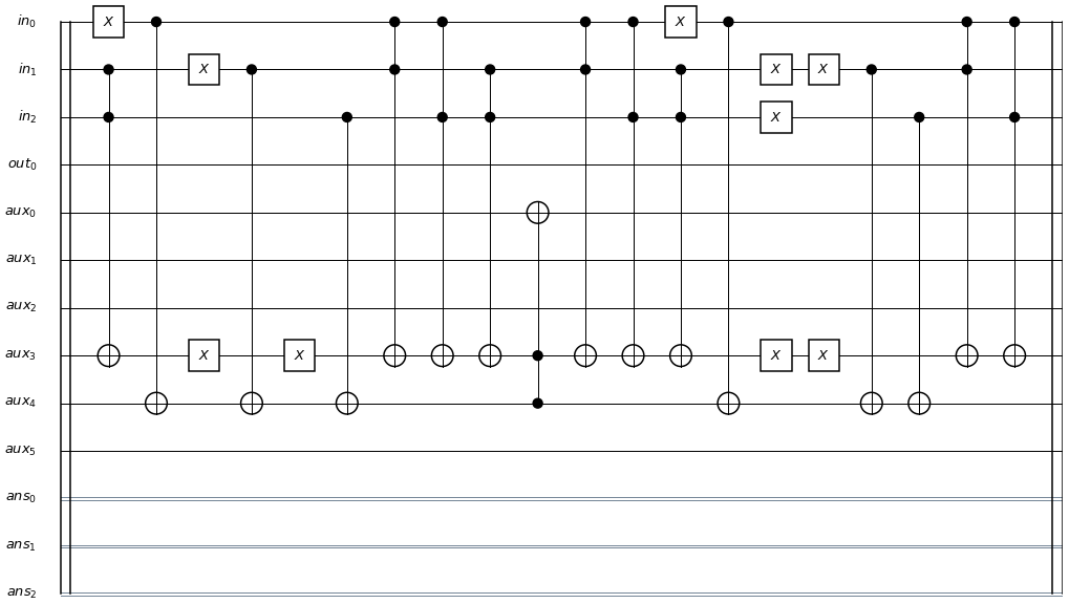
En este apartado se incluye un ejemplo del circuito generado por la primera implementación descrita en el apartado 4.3.2, cuya entrada es:

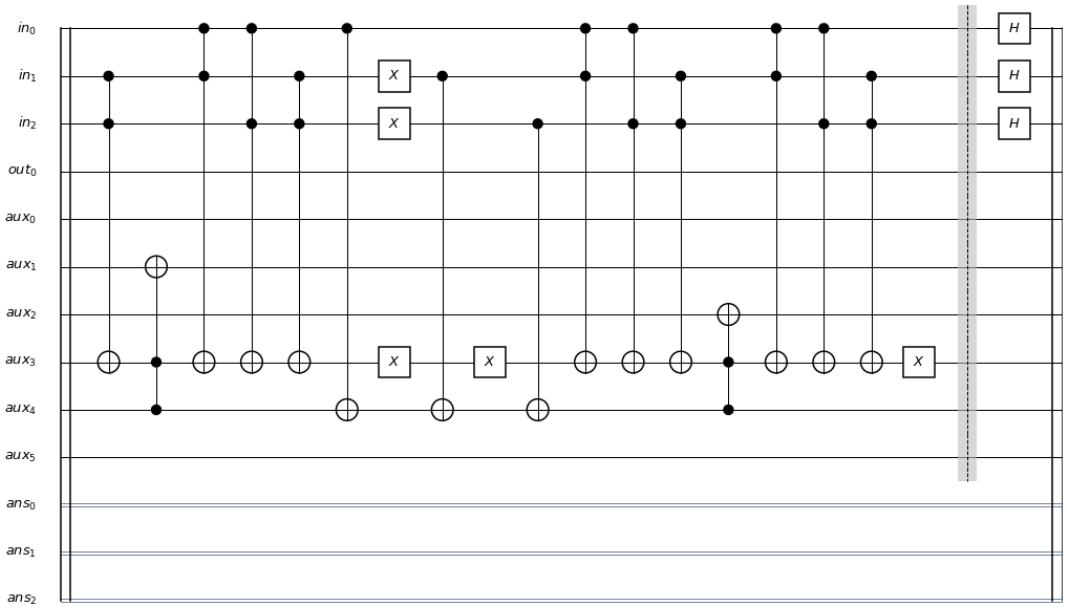
$$(x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee \neg x_3).$$

Los qubits señalados como **in**$_i$ son los correspondientes a la entrada, **out** a la salida, los **aux**$_i$ son los auxiliares y **ans**$_i$ son los registros utilizados para hacer las mediciones. Como el circuito es bastante extenso, se ha incluido una única iteración de Grover. Entre las dos primeras barreras se encuentra la parte del oráculo y entre las dos siguientes la inversión sobre la media.






in_0
in_1
in_2
out_0
aux_0
aux_1
aux_2
aux_3
aux_4
aux_5
ans_0
ans_1
ans_2


in_0
in_1
in_2
out_0
aux_0
aux_1
aux_2
aux_3
aux_4
aux_5
ans_0
ans_1
ans_2
H
H
H

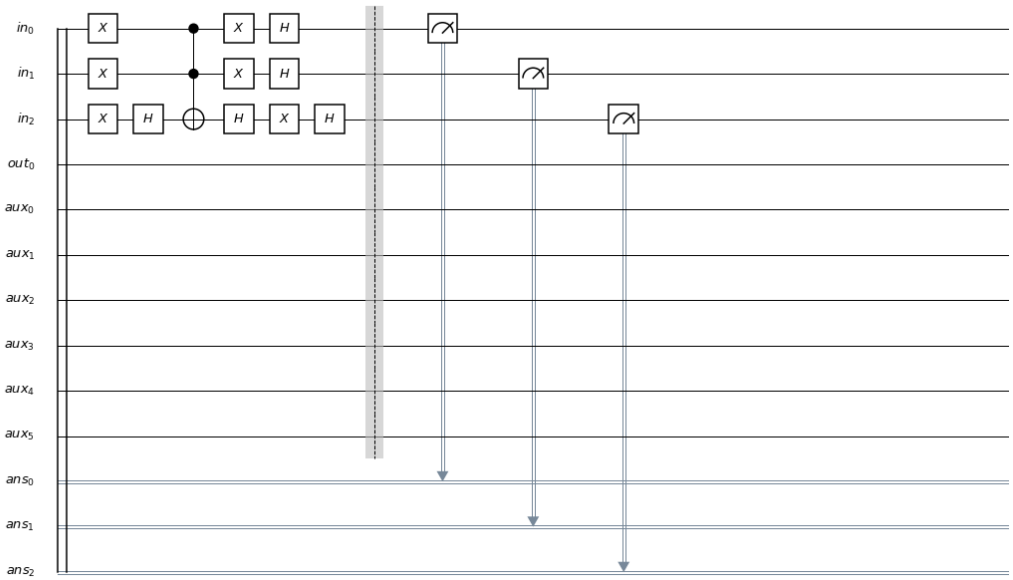


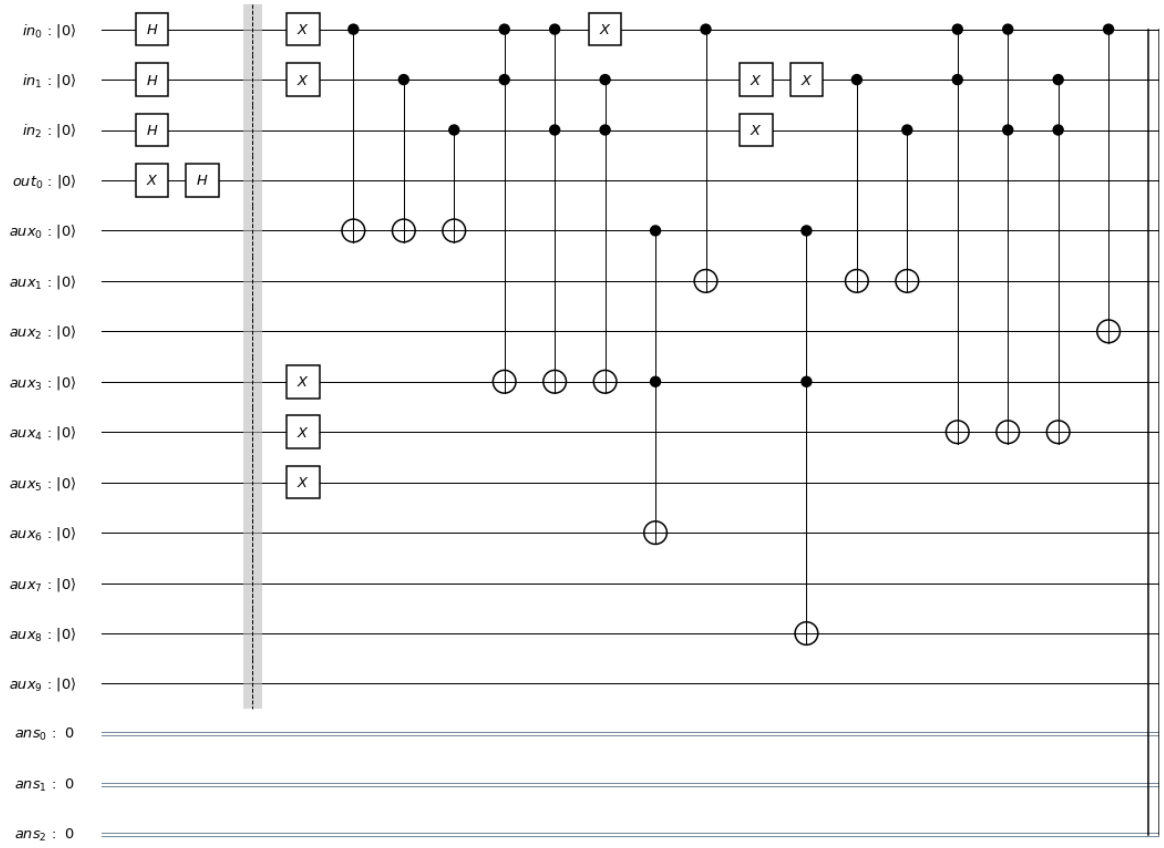
Figura A-1: Ejemplo de circuito cuántico realizado con Qiskit usando la primera implementación desarrollada. Por motivos de espacio esta figura está formada por las tres anteriores.

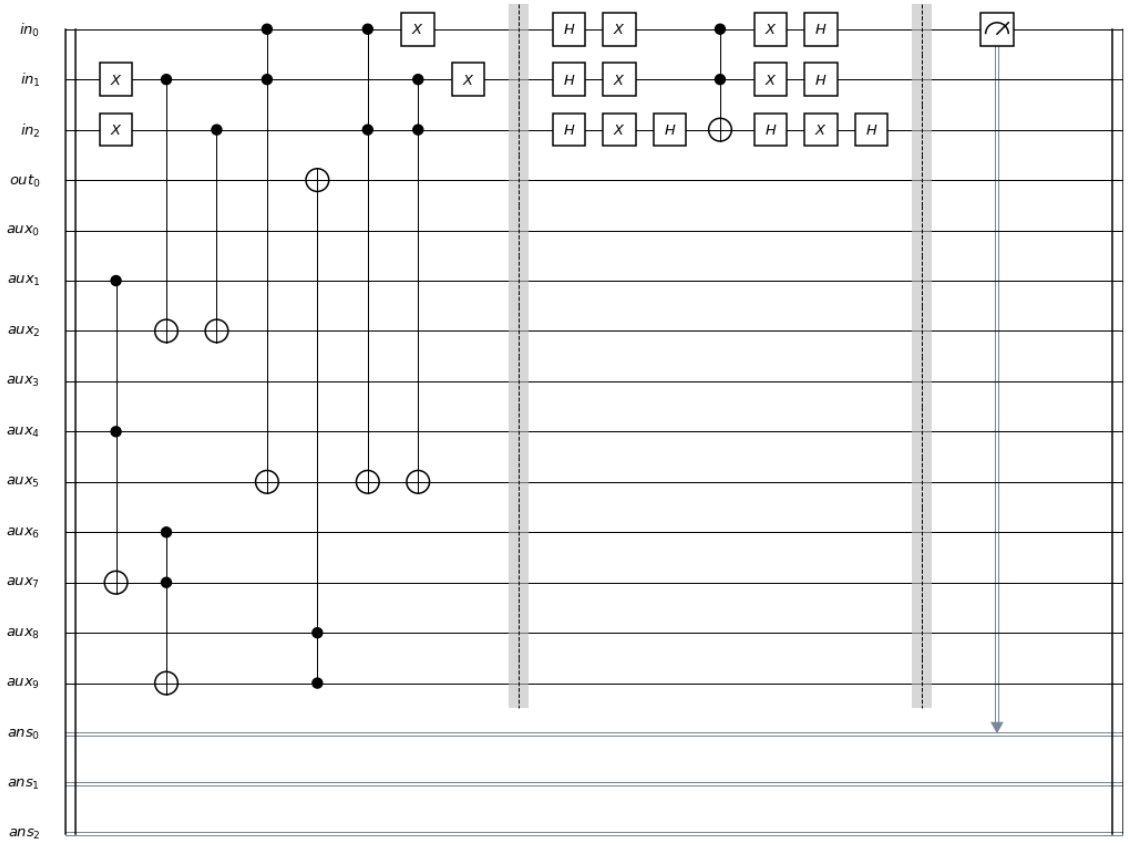
B Ejemplo de circuito de la segunda implementación

En este apartado se incluye un ejemplo del circuito generado por la primera implementación descrita en el apartado 4.3.2, cuya entrada es:

$$(x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee \neg x_3).$$

Los qubits señalados como **in**$_i$ son los correspondientes a la entrada, **out** a la salida, los **aux**$_i$ son los auxiliares y **ans**$_i$ son los registros utilizados para hacer las mediciones. Como el circuito es bastante extenso, se ha incluido una única iteración de Grover. Entre las dos primeras barreras se encuentra la parte del oráculo y entre las dos siguientes la inversión sobre la media.




in0
in1
in2
out0
aux0
aux1
aux2
aux3
aux4
aux5
aux6
aux7
aux8
aux9
ans0
ans1
ans2

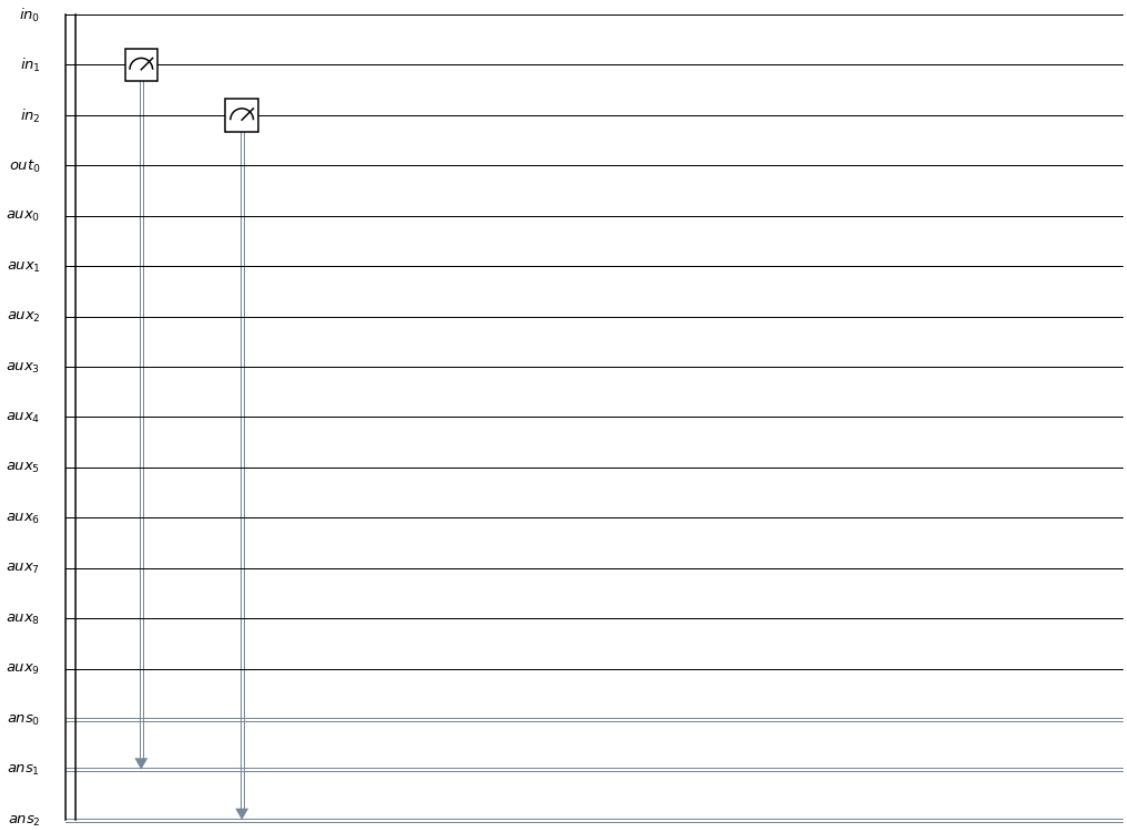


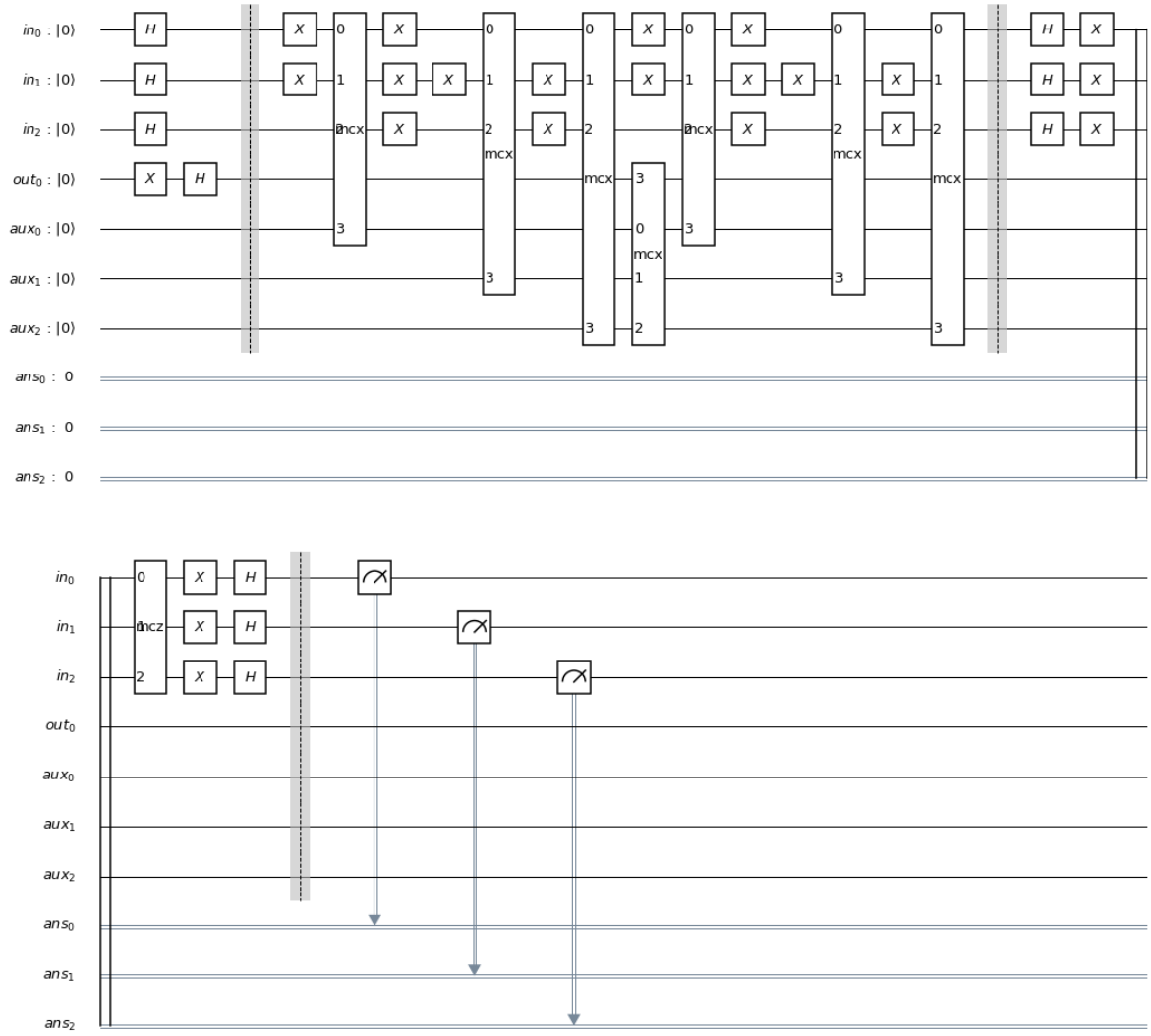
Figura B-1: Ejemplo de circuito cuántico realizado con Qiskit usando la segunda implementación desarrollada. Por motivos de espacio esta figura está formada por las tres anteriores.

C Ejemplo de circuito de la tercera implementación

En este apartado se incluye un ejemplo del circuito generado por la primera implementación descrita en el apartado 4.3.2, cuya entrada es:

$$(x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee \neg x_3).$$

Los qubits señalados como **in**$_i$ son los correspondientes a la entrada, **out** a la salida, los **aux**$_i$ son los auxiliares y **ans**$_i$ son los registros utilizados para hacer las mediciones. Como el circuito es bastante extenso, se ha incluido una única iteración de Grover. Entre las dos primeras barreras se encuentra la parte del oráculo y entre las dos siguientes la inversión sobre la media.



FiguraC-1: Ejemplo de circuito cuántico realizado con Qiskit usando la tercera implementación desarrollada.